Introduction

Problem Summary

Problem Statement

Despite the existence of many cyber security awareness programs, there is still a lack of effective, widespread cyber security training

As modern-day technology is ever evolving, the number of users who interact with technology on a daily basis increases consequently. As a result, the risk of an individual, or business, becoming a victim to cyber-crime increases proportionately. In particular, small and medium-sized businesses (SMBs) are the biggest sectors targeted by cyber-criminals [1], which stem from issues such as budget restraints and expressing a lack of understanding towards cyber security concepts.

In fact, as a consequence of COVID-19 changing the dynamic of industry standards this year, a statistical analysis from May 2020 (UK) showed that individuals experiencing targeted hacking increased by 77.41% - in comparison with the previous year [2]. This is most likely due to the fact that employees are encouraged to work from home via their personal computers. Consequently, this has fed into a new strategy whereby cyber-criminals are moving laterally into organisational infrastructure by targeting and infecting employees at their less secure personal computers [3]

In regards to this problem, this paper will explore the effectiveness of educational games - which has been shown to have an advantage on the learning outcome in comparison with traditional training material [4]. Therefore, this paper presents the following research question and hypothesis:

Research Question

Does teaching cyber security through a gamified medium improve user confidence in protecting against cyber attacks?

Hypothesis

Creating an educational cyber security game will leave users feeling more confident and aware with regards to cyber security concepts

Goals & Scope

The goal of this project is to investigate how to effectively apply gamification mechanics in order to teach cyber security principles appropriately. The expected result of this project is to create a multiplayer, online tabletop board game.

Literature Review & Research

Introduction

To date, this report includes an evaluation of literature review pertaining to pre-existing cyber security games, mechanics for game-based learning and current trends in cyber security training methodologies.

The problem with Current Cyber Security Training Programs

This report will specifically analyse the shortcomings and difficulties that relate to current training programs designed for small and medium-sized businesses [1, 5].

Problem	Description
1	SMBs can be heavily constrained by a limited budget
2	SMBs can be difficult to reach as they do not understand the severity of data breaches
3	SMBs are often distracted by the operational requirements for setting up and running a small business
4	SMBs struggle to identify their assets in terms of the risks associated with them

In regards to the delivery of training programs, providing generalised cyber security advice (from an independent advisor) has been shown to have little effect on changing the behaviour of employees within SMBs [1, 6]. Furthermore, traditional training simulations (including gamified video simulations) are generally undertaken in a formal environment which leads to a situation of recipients not absorbing the information well [2].

The methodology of cyber security exposure is also important as whilst employees may understand some general information about the vulnerability demonstrated, they may still fail to see how it relates to their workplace environment [5] or how they link together in a multifaceted social engineering attack [6]. This last point emphasises the need for widespread conceptual training in cyber security.

A Critical Analysis of Pre-Existing Cyber Security Games

Find some literature review on turn-based tactics style games if the final project leads that way?

For a full account of educational games and resources reviewed, please see Appendix A. My methodology for reviewing cyber security games was two fold:

- First - utilising the Google search engine with the following keywords: ‘cyber security’, ‘serious games’, ‘gamification’ and ‘game-based learning’ in order to look for any widely available games. These commonly returned web applications designed for students in all stages of education.
- Second - utilising Google Scholar and the IEEE Database with similar keywords in order to find academic papers which either reviewed other educational games, or were proposing one. For the games that were not available online, I summarised the key information and research results from the academic source material.

Building on from this, I concluded the following categories of educational game-based learning strategies in order to identify the most appropriate medium for the purpose of answering the research question and hypothesis established in this project.

3.3.1 Web Applications Advantages Disadvantages Simple point and click interactivity Can lack depth and relevance to a specific target (often designed for students) Easily accessible anywhere with an internet connection Not suitable for offline usage Cheap development cost & time 3.3.2 Video / Simulation Advantages Disadvantages Contextually appropriate for use within the workplace [5] Not appropriate for the general public & students Accessible both online & offline Requires multiple play-throughs if scenarios are divergent Typically not very fun as undertaken in a formal environment [5] 3.3.3 Cooperative Tabletop Advantages Disadvantages Encourages social engagement and team-working Requires multiple players Cheap to prototype and produce a physical implementation Requires much fine-tuning of rules and mechanics implemented Encourages thinking strategically 6 3.3.4 Task Management Advantages Disadvantages Easy to employ around current learning strategies (within the classroom or workplace) Requires long term evaluation of effectiveness Perfect example of procedural learning [7] Not a true application of an educational game 3.3.5 Single Player Advantages Disadvantages Immersive and engaging typically through story driven content High development cost & time Often places the player as a white / black hat hacker which encourages adversarial thinking Lack of exploration on how to prevent vulnerabilities as a target

Difficulties that Pre-Existing Cyber Security Games Face

Many of the educational games reviewed relied heavily on presenting facts and then subsequently quizzing the user with a related question. However, users can utilise common sense to rule out incorrect answers thus fail to invoke critical thinking and do not keep the user engaged.

As a solution, gamified strategies should incorporate a variety of factual, conceptual and procedural learning methodologies [7]. In particular, a strong conceptual understanding should be prioritised due to the rapidly changing landscape of cyber security in which cyber-criminals will always be engineering new attack vectors [6]. Therefore, it is imperative for end-users to be able to adapt their way of thinking when interacting with new technology.

For procedural learning, both Nova Cyber Lab [8] and Classcraft [9] (Appendix A) exemplify this by beginning with simple challenges and progressively increasing the difficulty of said challenges as the user progresses. Unlike the other games reviewed, Classcraft is unique as it encourages users within a team to continually expand upon their knowledge by working towards new goals and objectives collaboratively. Furthermore, this system incorporates real-world rewards and punishments to encourage user-engagement.

Finally, many serious cyber security games are designed primarily for university students and businesses but are not readily available to the general public [7]; this agrees with my own research - whereby many of the publicly accessible educational games I reviewed were considerably outdated and not intended for the general public.

Appropriate Gamification Mechanics

Summary of Requirements Outlined from Literature Review

Summarise the main requirements into a section / table

Forde's report [14] identifies the following gamification mechanics in order to increase the adoption rate of effective cyber security standards within the workplace:

• Avatar / User Profile • Badges / Privileges • Challenge • Competition • Collaboration • Feedback / Guidance • Goals / Objectives • Incentives / Rewards • Leaderboards • Points System • Progress / Levels • Role Playing • Story • Tips / Hints System

Why Use a Game - Based Learning Approach?

Resummarise these in a way that's relevant and influenced the decision choice for the final project and **more importantly,** how it relates to the background / problem summary.

As identified above, many challenges that arise when attempting to educate employees within the business environment occur due the formality of traditional training methodologies in which employees will aim to complete as fast as possible [6]. This absentmindedness often results in training content which is not absorbed effectively [1, 6]. However, these challenges can often be overcome with game-based learning strategies [4].

A summary of key strengths and motivations for gamification [4, 10, 11, 12, 13] include:

- Rapid progress paired with instant feedback - Strong user engagement

- Allows the user to learn from mistakes in a safe, non punishing environment which would otherwise discourage exploration (due to the fear of failure)

- Serves as a platform to encourage self-learning at the user's desired pace

- Allows the user to become deeply immersed into the 'game - world' where learning feels like a secondary objective

- Relatively cheap and low cost to produce in comparison to larger training schemes

- Requires little to no supervision

- Easy to distribute across multiple platforms and incorporate into the workplace

- Easy to integrate within events such as hackathons and other cyber security awareness gatherings

- Use of an integrated rewards system such as badges, hidden achievements and the desire to win

- Educationally appropriate by incorporating a structure/story that is contextually similar to the real world

Conclusion

To summarise, cyber security training programs need to be cost effective and target the specific needs and requirements of the workplace in order to be viable for SMBs. Furthermore, where traditional training methodologies fall short, gamified strategies can offer a more cost-effective solution for teaching relevant cyber security concepts. However, unless the game in question has a strong story (e.g. Hacknet [15]), many educational games reviewed included a number of flaws which stem from lack of gamified mechanics that encourage meaningful play. Forde's report [14] identifies these key mechanics as multiplayer, leaderboards, points system and competition which can be deemed as the most effective in motivating people to participate and learn about a subject that they would otherwise have little interest in. Lastly, there is a gap in the public domain in which an appropriately designed cyber security game could be of use.

Requirements for the Project

Commented [RB1]: Adapt this section from future tense to an overall evaluation and justification!

Introduction

Commented [RB2]: **"An analysis and specification of the solution to the problem"**

The following sections appropriately identify the stakeholder's who will be interacting with the game / application, the functional (and non-functional requirements) and finally refined user stories based off the requirements and identified stakeholders. These user stories are then categorised with a *MoSCoW* analysed and used in conjunction with the initial Gantt chart for the development of the project.

Stakeholder - Personas

From the initial literature review and research, It was clear that SMB's, employee training and students in training were the biggest areas in which pre-existing cyber security games were used – and are relevant. From this, the following appropriate stakeholders can be identified as:

- Employees
- Students (Higher Education)
- Security Consultants
- Security Advocates

Additionally, this project stands out in that it can be adapted to fit a certain scenario (in terms of units, map and the "story" present. This project will aim to deliver a Minimal Viable Product (MVP) using a secure web application – and the top 10 #OWASP vulnerabilities as a context. Therefore, Web/Software Developers can be added to the list of the potential stakeholders.

 [ref]	**Olivia** is an undergraduate Physics student with a strong interest in programming and cyber security. Although she does not major in Computer Science, she would still like to learn more about how to protect herself online and develop secure web applications. Olivia is also constantly frustrated with her own parents who have very limited experience when it comes to technology and the internet, and as such have succumbed to phishing emails – compromising their login details. Olivia hasn't had much luck in educating her dad, however her dad is an avid lover of strategy games such as Chess. In this scenario, Olivia could benefit from the application by playing out multiple scenarios with her father – who would be encouraged to learn the rules and information of each piece and thus becoming more aware to the technology around him. For this reason, the application should have a simple scenario which would explore the most common vulnerabilities related to human error.

[ref]

Ben is a software consultant for a medium-sized software consultancy business. He has a degree in Computer Science with a good awareness of current cyber security trends, however he is not an expert in cyber security.

However, as a consultant, Ben always wants to be able to develop software for his company's clients that is as secure as it is usable. Ben would like to host an activity which brings together all his employees (of varying experience) and participate in keeping up to date with cyber security trends, as well as the top vulnerabilities to look out for in all aspects of software development.

His requirements are free, and light weight training material as he does not have the required expertise, or funding to hire an expert security consultant.

To benefit his goal of secure software development, e.g. a web application, the project (game) will have a story scenario in which comprises the most common OWASP Top 10 attacks. The nature of multiplayer will encourage his employees to challenge each-other in a stimulating and memorable way, all whilst not contributing to requiring his employees to work anymore outside of their required duties.

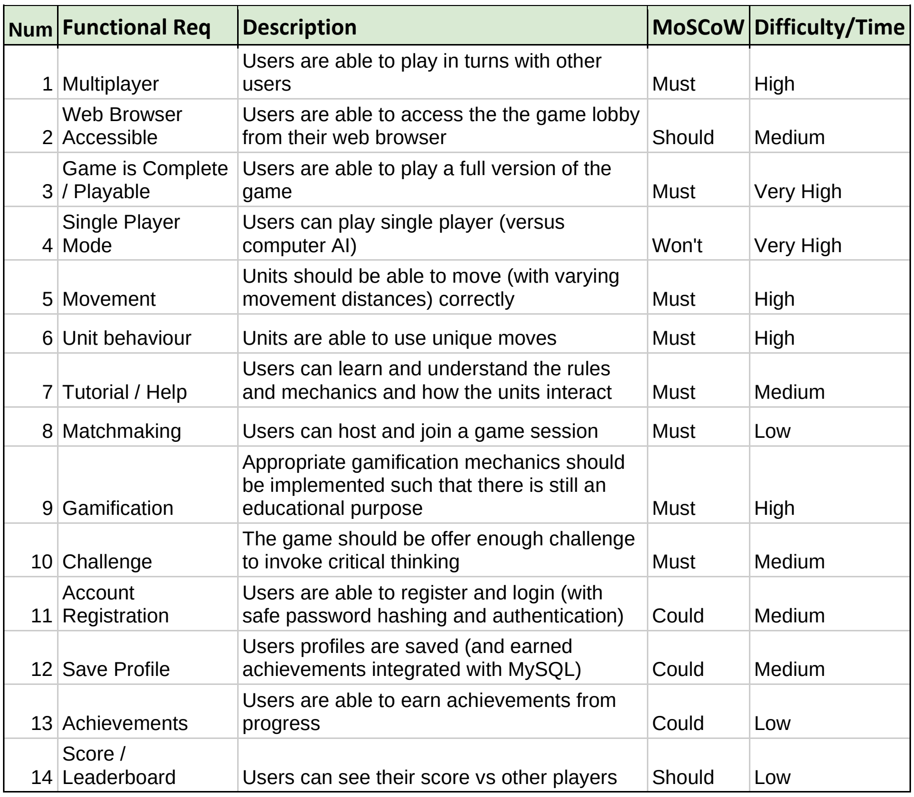
[ref]

Samira is a senior level cyber security consultant. As an expert in her field, her passion for cyber security extends to all her colleagues, friends and family. Samira completed her PhD on strategies to overcome the most common cyber-attacks. As a result, she has a strong interest in raising awareness and eliminating the human mistakes which most often lead to cyber vulnerabilities.

Although she does not have much experience with video games, she does however host a games night once a month in which her friends and family come round to play board games and watch movies. Although many of these games are abstract from reality, Samira really appreciates the value of educational games in which her and her friends have all learned something valuable whilst playing (such as quiz games on Geography, Culture, Science and Trivia).

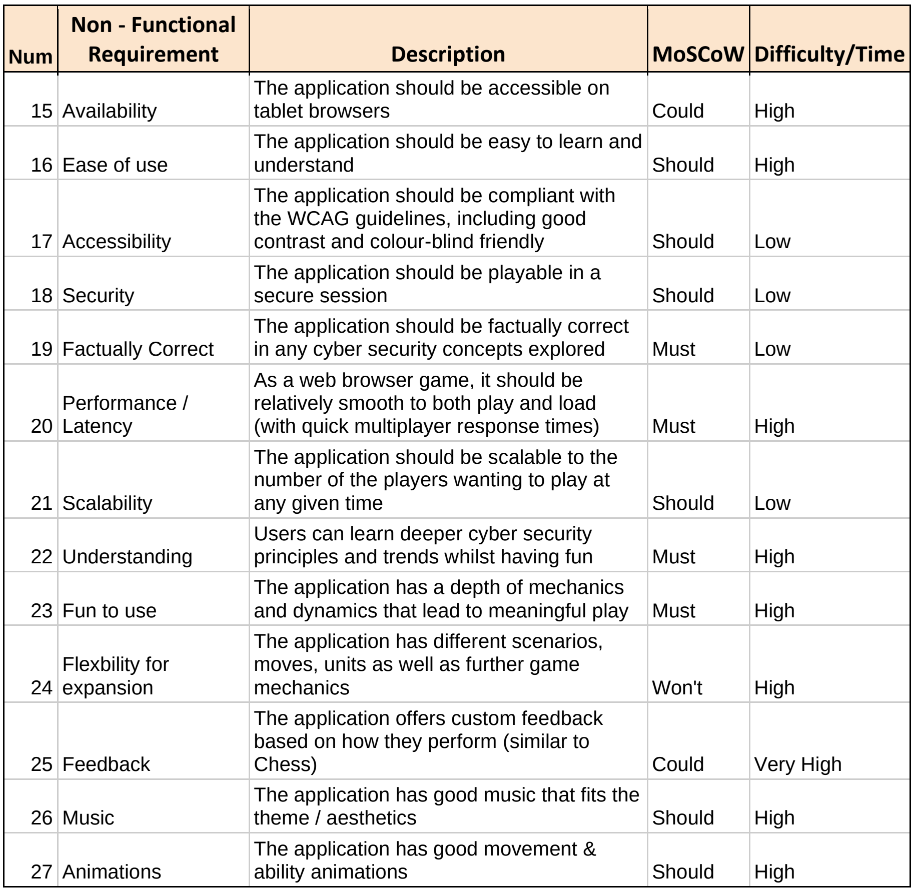
In this scenario, Samira would benefit from the application by being able to run a small competitive scenario in pairs with her friends, all whilst keeping work outside of her social life, but very much including her passion still.

Functional Requirements



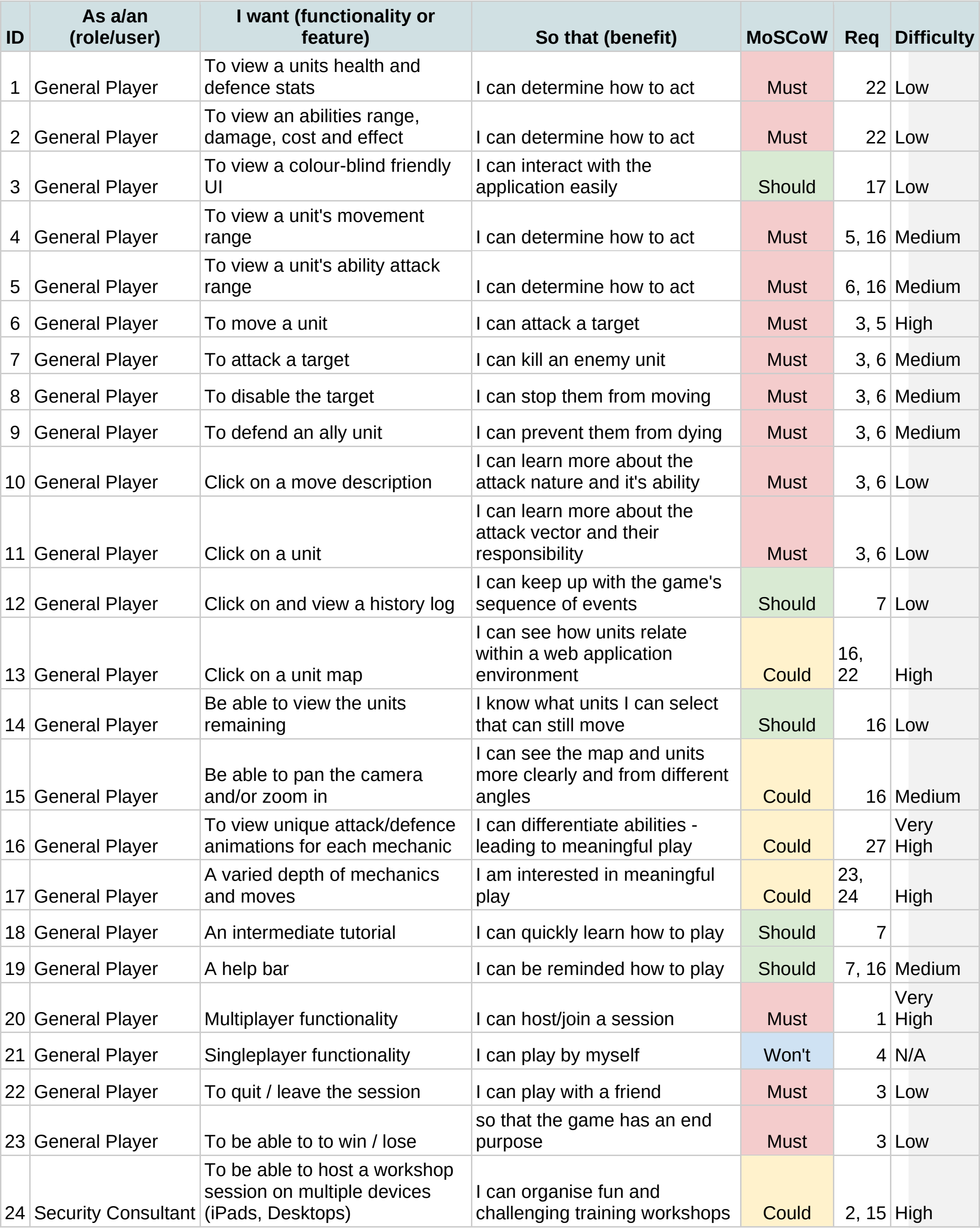
Num	Functional Req	Description	MoSCoW	Difficulty/Time
1	Multiplayer	Users are able to play in turns with other users	Must	High
2	Web Browser Accessible	Users are able to access the the game lobby from their web browser	Should	Medium
3	Game is Complete / Playable	Users are able to play a full version of the game	Must	Very High
4	Single Player Mode	Users can play single player (versus computer AI)	Won't	Very High
5	Movement	Units should be able to move (with varying movement distances) correctly	Must	High
6	Unit behaviour	Units are able to use unique moves	Must	High
7	Tutorial / Help	Users can learn and understand the rules and mechanics and how the units interact	Must	Medium
8	Matchmaking	Users can host and join a game session	Must	Low
9	Gamification	Appropriate gamification mechanics should be implemented such that there is still an educational purpose	Must	High
10	Challenge	The game should be offer enough challenge to invoke critical thinking	Must	Medium
11	Account Registration	Users are able to register and login (with safe password hashing and authentication)	Could	Medium
12	Save Profile	Users profiles are saved (and earned achievements integrated with MySQL)	Could	Medium
13	Achievements	Users are able to earn achievements from progress	Could	Low
14	Score / Leaderboard	Users can see their score vs other players	Should	Low

Non - Functional Requirements



Num	Non - Functional Requirement	Description	MoSCoW	Difficulty/Time
15	Availability	The application should be accessible on tablet browsers	Could	High
16	Ease of use	The application should be easy to learn and understand	Should	High
17	Accessibility	The application should be compliant with the WCAG guidelines, including good contrast and colour-blind friendly	Should	Low
18	Security	The application should be playable in a secure session	Should	Low
19	Factually Correct	The application should be factually correct in any cyber security concepts explored	Must	Low
20	Performance / Latency	As a web browser game, it should be relatively smooth to both play and load (with quick multiplayer response times)	Must	High
21	Scalability	The application should be scalable to the number of the players wanting to play at any given time	Should	Low
22	Understanding	Users can learn deeper cyber security principles and trends whilst having fun	Must	High
23	Fun to use	The application has a depth of mechanics and dynamics that lead to meaningful play	Must	High
24	Flexbility for expansion	The application has different scenarios, moves, units as well as further game mechanics	Won't	High
25	Feedback	The application offers custom feedback based on how they perform (similar to Chess)	Could	Very High
26	Music	The application has good music that fits the theme / aesthetics	Should	High
27	Animations	The application has good movement & ability animations	Should	High

User Stories

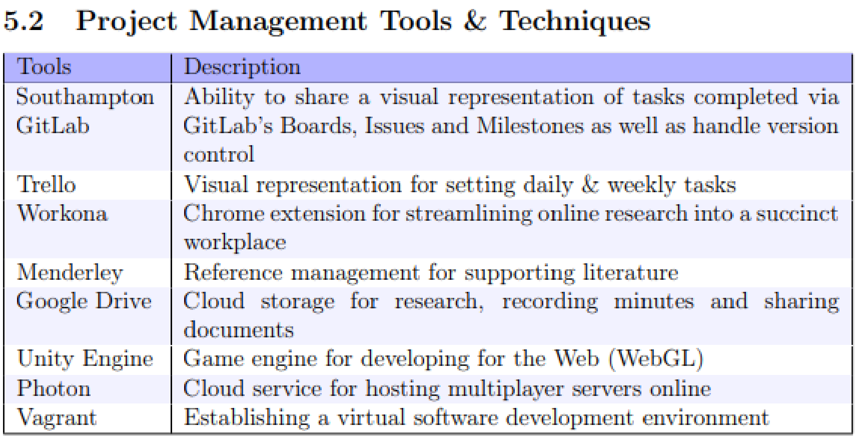


ID	As a/an (role/user)	I want (functionality or feature)	So that (benefit)	MoSCoW	Req	Difficulty
1	General Player	To view a units health and defence stats	I can determine how to act	Must	22	Low
2	General Player	To view an abilities range, damage, cost and effect	I can determine how to act	Must	22	Low
3	General Player	To view a colour-blind friendly UI	I can interact with the application easily	Should	17	Low
4	General Player	To view a unit's movement range	I can determine how to act	Must	5, 16	Medium
5	General Player	To view a unit's ability attack range	I can determine how to act	Must	6, 16	Medium
6	General Player	To move a unit	I can attack a target	Must	3, 5	High
7	General Player	To attack a target	I can kill an enemy unit	Must	3, 6	Medium
8	General Player	To disable the target	I can stop them from moving	Must	3, 6	Medium
9	General Player	To defend an ally unit	I can prevent them from dying	Must	3, 6	Medium
10	General Player	Click on a move description	I can learn more about the attack nature and it's ability	Must	3, 6	Low
11	General Player	Click on a unit	I can learn more about the attack vector and their responsibility	Must	3, 6	Low
12	General Player	Click on and view a history log	I can keep up with the game's sequence of events	Should	7	Low
13	General Player	Click on a unit map	I can see how units relate within a web application environment	Could	16, 22	High
14	General Player	Be able to view the units remaining	I know what units I can select that can still move	Should	16	Low
15	General Player	Be able to pan the camera and/or zoom in	I can see the map and units more clearly and from different angles	Could	16	Medium
16	General Player	To view unique attack/defence animations for each mechanic	I can differentiate abilities - leading to meaningful play	Could	27	Very High
17	General Player	A varied depth of mechanics and moves	I am interested in meaningful play	Could	23, 24	High
18	General Player	An intermediate tutorial	I can quickly learn how to play	Should	7	
19	General Player	A help bar	I can be reminded how to play	Should	7, 16	Medium
20	General Player	Multiplayer functionality	I can host/join a session	Must	1	Very High
21	General Player	Singleplayer functionality	I can play by myself	Won't	4	N/A
22	General Player	To quit / leave the session	I can play with a friend	Must	3	Low
23	General Player	To be able to to win / lose	so that the game has an end purpose	Must	3	Low
24	Security Consultant	To be able to host a workshop session on multiple devices (iPads, Desktops)	I can organise fun and challenging training workshops	Could	2, 15	High

25	Student	To learn about the OWASP #10	I can learn more about the most popular attacks and mitigations for them	Must	22	Low
26	General Player	Purchase & upgrade new units and abilities	I have a deeper reason to keep playing	Could	24	Very High
27	General Player	To register an account and login	I can view and save my unlocked units	Could	11, 12	High

Project Management Tools Utilised

Tools	Description
Southampton GitLab	Ability to share a visual representation of tasks completed via GitLab's Boards, Issues and Milestones as well as handle version control
Trello	Visual representation for setting daily & weekly tasks
Workona	Chrome extension for streamlining online research into a succinct workplace
Menderley	Reference management for supporting literature
Google Drive	Cloud storage for research, recording minutes and sharing documents
Unity Engine	Game engine for developing for the Web (WebGL)
Photon Unity Networking	Cloud service for hosting multiplayer servers online



5.2 Project Management Tools & Techniques

Tools	Description
Southampton GitLab	Ability to share a visual representation of tasks completed via GitLab's Boards, Issues and Milestones as well as handle version control
Trello	Visual representation for setting daily & weekly tasks
Workona	Chrome extension for streamlining online research into a succinct workplace
Menderley	Reference management for supporting literature
Google Drive	Cloud storage for research, recording minutes and sharing documents
Unity Engine	Game engine for developing for the Web (WebGL)
Photon	Cloud service for hosting multiplayer servers online
Vagrant	Establishing a virtual software development environment

Constraints

Commented [RB3]: •Add in some content here about some constraints which will make the project particularly difficult, or need to be foregone with a justification

- The game has to be efficient to be able to run in low memory on a web browser
- HTML /WebGL and different browsers on different devices have different limitations on rendering ability, screen real estate, input (such as touch screen vs clicking)
- Given the disruption of Covid19, creating a physical paper / cardboard prototypes and testing before development will not be possible hence it will be difficult to ensure we are building a “right” MVP
- Given the limited time duration, it wont realistically be possible to design an iterative game in which I would have time to retrieve ethics ergo approval, get feedback from playtesting, then redesign/rebuild on that prototype. Later on this is a core part of the expanded MDA framework in which this would be an incredibly useful part of game development
- Also given the time development of this project, only one “scenario” example will be created, however following this, there should be the archiectecture to quickly use the game as a framework to build different scenarios, units and moves and just be changing the parameters, content and applying different parameters. A lot of development time goes both into the coding and understanding (to build a flexible decoupled game) as well as the aesthetics and design

Risk Assessment?

Updated Gantt Charts?

Design of the Project

Introduction

Commented [RB4]: •Intro about the goals and objectives of said project, the overall design and why

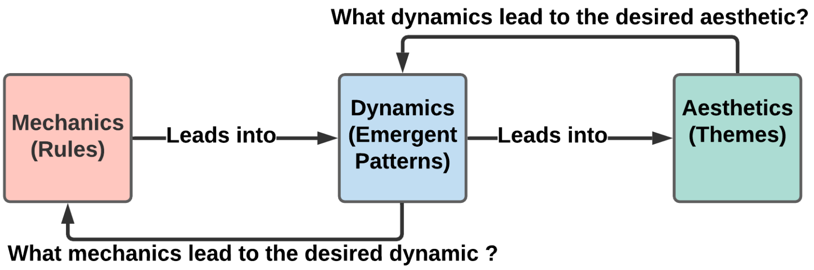
Based on the requirements (…) outlined above, multiplayer and challenge are two major themes which also tie into the desired aesthetics outlined within the MDA framework (explained below). As a result, a turn-based strategy (similar to Chess) seemed like an appropriate player-vs-player game. Furthermore, taking inspiration from Final Fantasy Tactics and Into the Breach – which uses mecha units with certain abilities, I really wanted to illustrate cyber security attacks and defences in a format similar to this as both the theme was appropriate and the games have had much success in both critics and reviews.

This section introduces the MDA framework which outlines the mechanics, dynamics and aesthetics of the game, the cyber security cards (illustrated by the OWASP #10) and finally the overall design of the game (in terms of user experience and accessibility).

Applying the MDA Framework

The MDA (Mechanics, Dynamics, Aesthetics) framework is the most popular framework in order to identify the requirements of building a game. It is an iterative process in which you outline your mechanics, then from this the dynamics that gameplay loop and finally the aesthetics that you wish to achieve. You can then revise and tweak the mechanics in order to create better dynamics that fit the aesthetics.

Fig X – MDA Framework Visualised


What dynamics lead to the desired aesthetic?
Mechanics (Rules)
Leads into
Dynamics (Emergent Patterns)
Leads into
Aesthetics (Themes)
What mechanics lead to the desired dynamic ?

Mechanics

The mechanics are fundamentally the most basic rules and actions a player can do within the Gameworld

- Action points being required for certain moves
- Moves have a distance range and can be blocked by other units
- Movement is in a BFS outwards and can be blocked by both enemy and player units
- Units can only attack once and move once per turn, but a move can be multi-targeting hitting
- Units can do damage, bypass shields, restore defence and disable another unit from acting for a turn
- Each movable unit has 2 abilities, whereas the static units (webserver/database) cannot be interacted with

Dynamics

The dynamics can be thought of as the emergent behaviour that arises from gameplay when the mechanics are implemented [ref].

- Players may choose to target the webserver/database first to reduce AP gain
- Players may use shield destroying moves on heavy units
- Players may use cheaper moves to save/bank AP (and potentially retreat)
- Players may shield their database/webserver in order to protect them
- Players may use the scout unit + bypass defence move to target units with a lot of defence but little HP (such as the xx server)

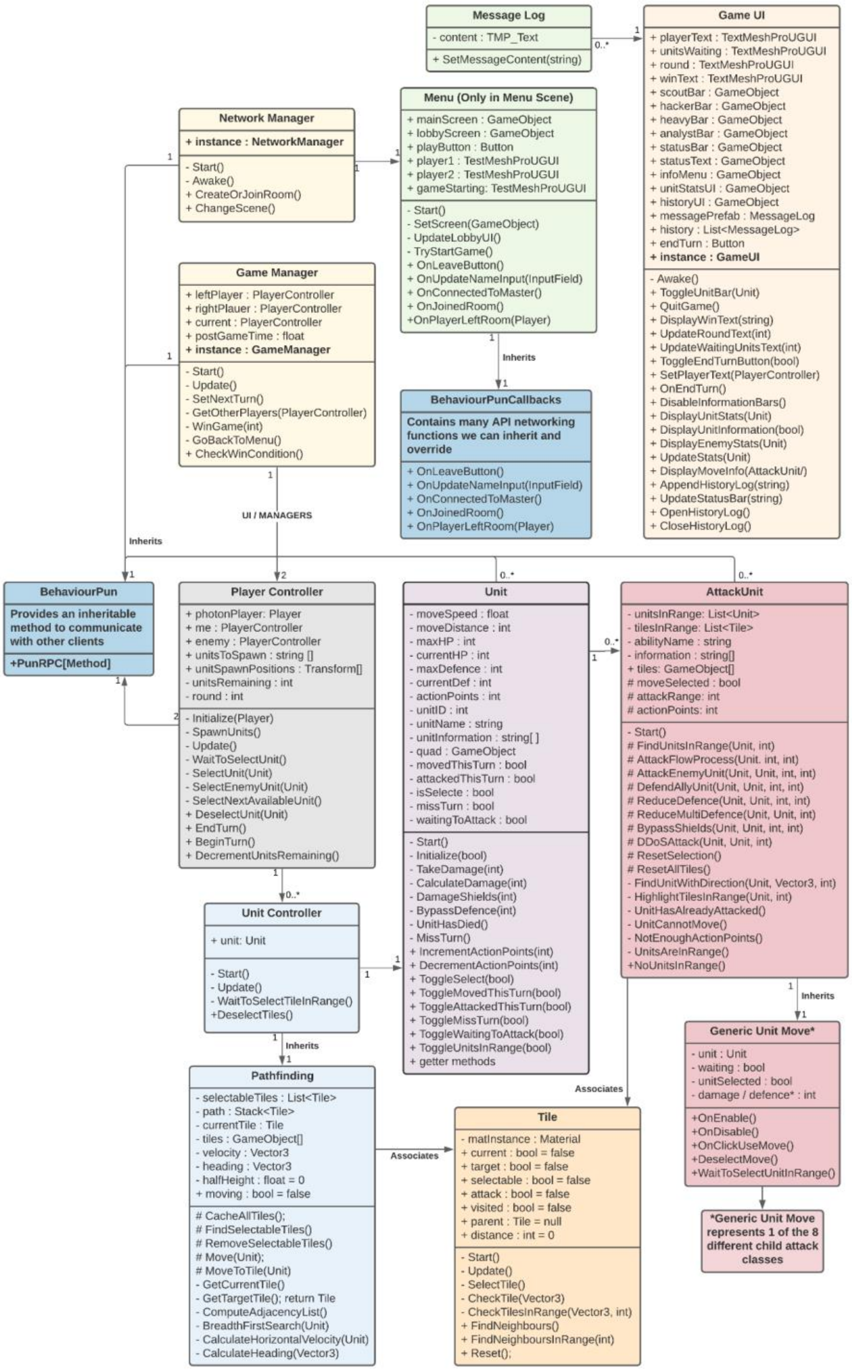
Aesthetics

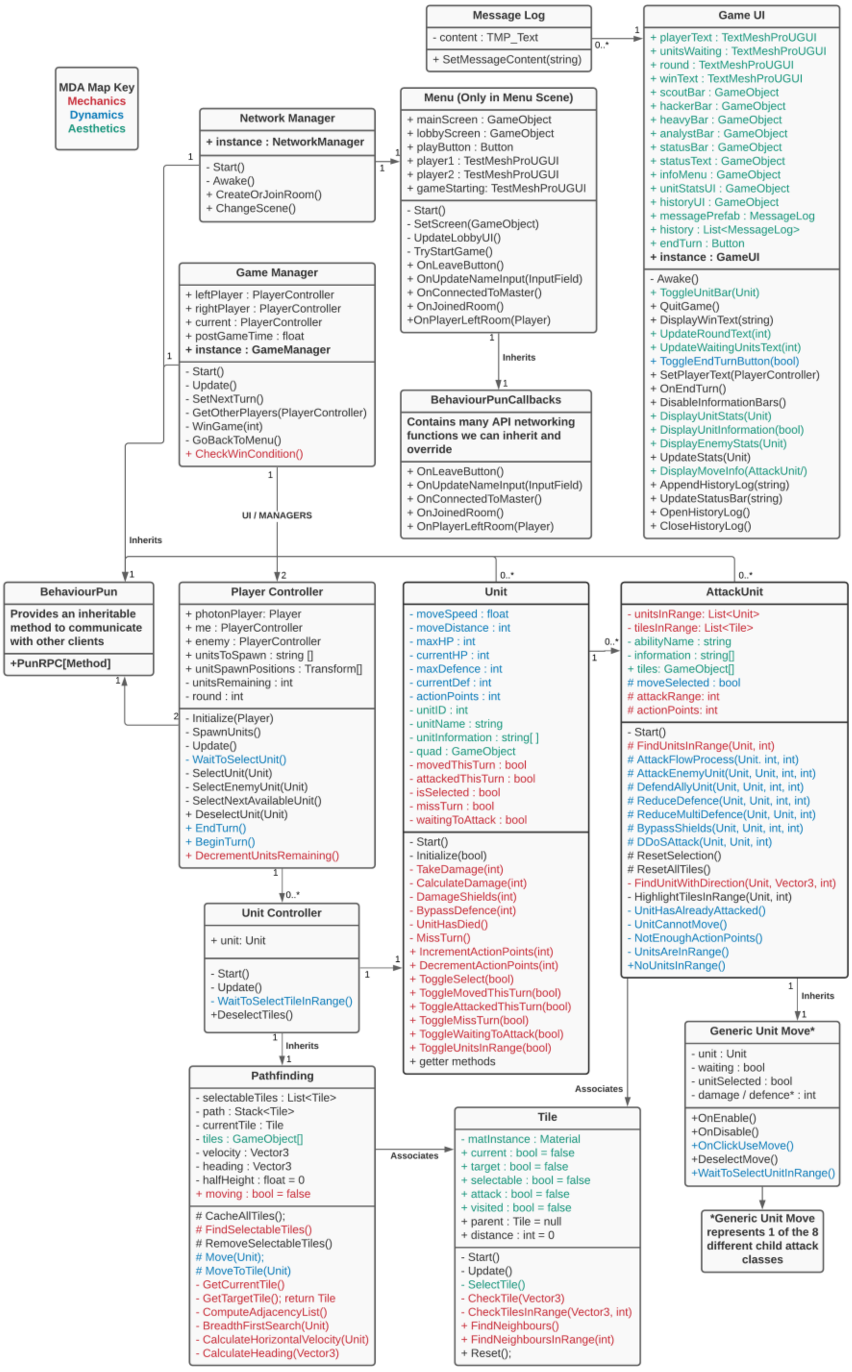
The aesthetics can be thought of as “the emotional response a game should illicit from the player” [ref?].

They can be broken down into:

- Sensations (e.g. emotional games)
- Fantasy (immersion)
- Narrative (story-rich)
- **Challenge (puzzles, obstacles)**
- **Fellowship (co-operative or multiplayer)**
- Discovery (open world)
- Expression (Character creation, sandbox building)
- Submission (farming simulators)

Map these to the class diagram!

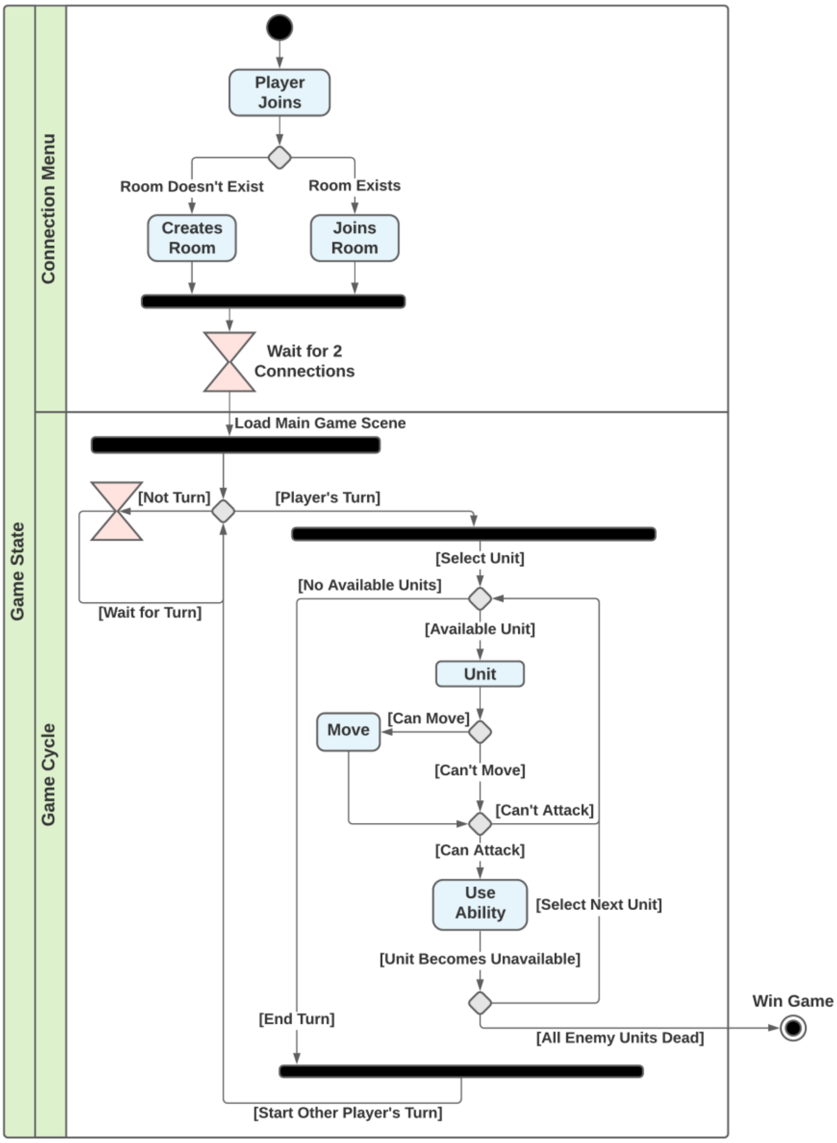

Message Log
- content : TMP_Text
+ SetMessageContent(string)
0..*
1
Game UI
+ playerText : TextMeshProUGUI
+ unitsWaiting : TextMeshProUGUI
+ round : TextMeshProUGUI
+ winText : TextMeshProUGUI
+ scoutBar : GameObject
+ hackerBar : GameObject
+ heavyBar : GameObject
+ analystBar : GameObject
+ statusBar : GameObject
+ statusText : GameObject
+ infoMenu : GameObject
+ unitStatsUI : GameObject
+ historyUI : GameObject
+ messagePrefab : MessageLog
+ history : List<MessageLog>
+ endTurn : Button
+ instance : GameUI
- Awake()
+ ToggleUnitBar(Unit)
+ QuitGame()
+ DisplayWinText(string)
+ UpdateRoundText(int)
+ UpdateWaitingUnitsText(int)
+ ToggleEndTurnButton(bool)
+ SetPlayerText(PlayerController)
+ OnEndTurn()
+ DisableInformationBars()
+ DisplayUnitStats(Unit)
+ DisplayUnitInformation(bool)
+ DisplayEnemyStats(Unit)
+ UpdateStats(Unit)
+ DisplayMoveInfo(AttackUnit/)
+ AppendHistoryLog(string)
+ UpdateStatusBar(string)
+ OpenHistoryLog()
+ CloseHistoryLog()
Network Manager
+ instance : NetworkManager
- Start()
- Awake()
+ CreateOrJoinRoom()
+ ChangeScene()
1
1
1
Menu (Only in Menu Scene)
+ mainScreen : GameObject
+ lobbyScreen : GameObject
+ playButton : Button
+ player1 : TestMeshProUGUI
+ player2 : TestMeshProUGUI
+ gameStarting: TestMeshProUGUI
- Start()
- SetScreen(GameObject)
- UpdateLobbyUI()
- TryStartGame()
+ OnLeaveButton()
+ OnUpdateNameInput(InputField)
+ OnConnectedToMaster()
+ OnJoinedRoom()
+OnPlayerLeftRoom(Player)
1
Inherits
1
BehaviourPunCallbacks
Contains many API networking functions we can inherit and override
+ OnLeaveButton()
+ OnUpdateNameInput(InputField)
+ OnConnectedToMaster()
+ OnJoinedRoom()
+ OnPlayerLeftRoom(Player)
Game Manager
+ leftPlayer : PlayerController
+ rightPlauer : PlayerController
+ current : PlayerController
+ postGameTime : float
+ instance : GameManager
- Start()
- Update()
- SetNextTurn()
- GetOtherPlayers(PlayerController)
- WinGame(int)
- GoBackToMenu()
+ CheckWinCondition()
1
1
UI / MANAGERS
Inherits
1
2
0..*
0..*
BehaviourPun
Provides an inheritable method to communicate with other clients
+PunRPC[Method]
1
2
Player Controller
+ photonPlayer: Player
+ me : PlayerController
+ enemy : PlayerController
+ unitsToSpawn : string []
+ unitSpawnPositions : Transform[]
- unitsRemaining : int
- round : int
- Initialize(Player)
- SpawnUnits()
- Update()
- WaitToSelectUnit()
- SelectUnit(Unit)
- SelectEnemyUnit(Unit)
- SelectNextAvailableUnit()
+ DeselectUnit(Unit)
+ EndTurn()
+ BeginTurn()
+ DecrementUnitsRemaining()
Unit
- moveSpeed : float
- moveDistance : int
- maxHP : int
- currentHP : int
- maxDefence : int
- currentDef : int
- actionPoints : int
- unitID : int
- unitName : string
- unitInformation : string[ ]
- quad : GameObject
- movedThisTurn : bool
- attackedThisTurn : bool
- isSelecte : bool
- missTurn : bool
- waitingToAttack : bool
- Start()
- Initialize(bool)
- TakeDamage(int)
- CalculateDamage(int)
- DamageShields(int)
- BypassDefence(int)
- UnitHasDied()
- MissTurn()
+ IncrementActionPoints(int)
+ DecrementActionPoints(int)
+ ToggleSelect(bool)
+ ToggleMovedThisTurn(bool)
+ ToggleAttackedThisTurn(bool)
+ ToggleMissTurn(bool)
+ ToggleWaitingToAttack(bool)
+ ToggleUnitsInRange(bool)
+ getter methods
0..*
1
AttackUnit
- unitsInRange: List<Unit>
- tilesInRange: List<Tile>
- abilityName : string
- information : string[]
+ tiles: GameObject[]
# moveSelected : bool
# attackRange: int
# actionPoints: int
- Start()
# FindUnitsInRange(Unit, int)
# AttackFlowProcess(Unit. int, int)
# AttackEnemyUnit(Unit, Unit, int, int)
# DefendAllyUnit(Unit, Unit, int, int)
# ReduceDefence(Unit, Unit, int, int)
# ReduceMultiDefence(Unit, Unit, int)
# BypassShields(Unit, Unit, int, int)
# DDoSAttack(Unit, Unit, int)
# ResetSelection()
# ResetAllTiles()
- FindUnitWithDirection(Unit, Vector3, int)
- HighlightTilesInRange(Unit, int)
- UnitHasAlreadyAttacked()
- UnitCannotMove()
- NotEnoughActionPoints()
- UnitsAreInRange()
+NoUnitsInRange()
1
0..*
Unit Controller
+ unit: Unit
- Start()
- Update()
- WaitToSelectTileInRange()
+DeselectTiles()
1
1
1
Inherits
1
1
Inherits
1
Generic Unit Move*
- unit : Unit
- waiting : bool
- unitSelected : bool
- damage / defence* : int
+OnEnable()
+OnDisable()
+OnClickUseMove()
+DeselectMove()
+WaitToSelectUnitInRange()
*Generic Unit Move represents 1 of the 8 different child attack classes
Pathfinding
- selectableTiles : List<Tile>
- path : Stack<Tile>
- currentTile : Tile
- tiles : GameObject[]
- velocity : Vector3
- heading : Vector3
- halfHeight : float = 0
+ moving : bool = false
# CacheAllTiles();
# FindSelectableTiles()
# RemoveSelectableTiles()
# Move(Unit);
# MoveToTile(Unit)
- GetCurrentTile()
- GetTargetTile(); return Tile
- ComputeAdjacencyList()
- BreadthFirstSearch(Unit)
- CalculateHorizontalVelocity(Unit)
- CalculateHeading(Vector3)
Associates
Associates
Tile
- matInstance : Material
+ current : bool = false
+ target : bool = false
+ selectable : bool = false
+ attack : bool = false
+ visited : bool = false
+ parent : Tile = null
+ distance : int = 0
- Start()
- Update()
- SelectTile()
- CheckTile(Vector3)
- CheckTilesInRange(Vector3, int)
+ FindNeighbours()
+ FindNeighboursInRange(int)
+ Reset();


MDA Map Key
Mechanics
Dynamics
Aesthetics
Message Log
- content : TMP_Text
+ SetMessageContent(string)
0..*
1
Game UI
+ playerText : TextMeshProUGUI
+ unitsWaiting : TextMeshProUGUI
+ round : TextMeshProUGUI
+ winText : TextMeshProUGUI
+ scoutBar : GameObject
+ hackerBar : GameObject
+ heavyBar : GameObject
+ analystBar : GameObject
+ statusBar : GameObject
+ statusText : GameObject
+ infoMenu : GameObject
+ unitStatsUI : GameObject
+ historyUI : GameObject
+ messagePrefab : MessageLog
+ history : List<MessageLog>
+ endTurn : Button
+ instance : GameUI
- Awake()
+ ToggleUnitBar(Unit)
+ QuitGame()
+ DisplayWinText(string)
+ UpdateRoundText(int)
+ UpdateWaitingUnitsText(int)
+ ToggleEndTurnButton(bool)
+ SetPlayerText(PlayerController)
+ OnEndTurn()
+ DisableInformationBars()
+ DisplayUnitStats(Unit)
+ DisplayUnitInformation(bool)
+ DisplayEnemyStats(Unit)
+ UpdateStats(Unit)
+ DisplayMoveInfo(AttackUnit/)
+ AppendHistoryLog(string)
+ UpdateStatusBar(string)
+ OpenHistoryLog()
+ CloseHistoryLog()
Network Manager
+ instance : NetworkManager
- Start()
- Awake()
+ CreateOrJoinRoom()
+ ChangeScene()
1
1
1
Menu (Only in Menu Scene)
+ mainScreen : GameObject
+ lobbyScreen : GameObject
+ playButton : Button
+ player1 : TestMeshProUGUI
+ player2 : TestMeshProUGUI
+ gameStarting: TestMeshProUGUI
- Start()
- SetScreen(GameObject)
- UpdateLobbyUI()
- TryStartGame()
+ OnLeaveButton()
+ OnUpdateNameInput(InputField)
+ OnConnectedToMaster()
+ OnJoinedRoom()
+OnPlayerLeftRoom(Player)
1
Inherits
1
BehaviourPunCallbacks
Contains many API networking functions we can inherit and override
+ OnLeaveButton()
+ OnUpdateNameInput(InputField)
+ OnConnectedToMaster()
+ OnJoinedRoom()
+ OnPlayerLeftRoom(Player)
Game Manager
+ leftPlayer : PlayerController
+ rightPlayer : PlayerController
+ current : PlayerController
+ postGameTime : float
+ instance : GameManager
- Start()
- Update()
- SetNextTurn()
- GetOtherPlayers(PlayerController)
- WinGame(int)
- GoBackToMenu()
+ CheckWinCondition()
1
1
UI / MANAGERS
Inherits
1
2
0..*
0..*
BehaviourPun
Provides an inheritable method to communicate with other clients
+PunRPC[Method]
1
2
Player Controller
+ photonPlayer: Player
+ me : PlayerController
+ enemy : PlayerController
+ unitsToSpawn : string []
+ unitSpawnPositions : Transform[]
- unitsRemaining : int
- round : int
- Initialize(Player)
- SpawnUnits()
- Update()
- WaitToSelectUnit()
- SelectUnit(Unit)
- SelectEnemyUnit(Unit)
- SelectNextAvailableUnit()
+ DeselectUnit(Unit)
+ EndTurn()
+ BeginTurn()
+ DecrementUnitsRemaining()
1
0..*
Unit Controller
+ unit: Unit
- Start()
- Update()
- WaitToSelectTileInRange()
+DeselectTiles()
1
1
1
Inherits
1
Pathfinding
- selectableTiles : List<Tile>
- path : Stack<Tile>
- currentTile : Tile
- tiles : GameObject[]
- velocity : Vector3
- heading : Vector3
- halfHeight : float = 0
+ moving : bool = false
# CacheAllTiles();
# FindSelectableTiles()
# RemoveSelectableTiles()
# Move(Unit);
# MoveToTile(Unit)
- GetCurrentTile()
- GetTargetTile(); return Tile
- ComputeAdjacencyList()
- BreadthFirstSearch(Unit)
- CalculateHorizontalVelocity(Unit)
- CalculateHeading(Vector3)
Associates
Unit
- moveSpeed : float
- moveDistance : int
- maxHP : int
- currentHP : int
- maxDefence : int
- currentDef : int
- actionPoints : int
- unitID : int
- unitName : string
- unitInformation : string[ ]
- quad : GameObject
- movedThisTurn : bool
- attackedThisTurn : bool
- isSelected : bool
- missTurn : bool
- waitingToAttack : bool
- Start()
- Initialize(bool)
- TakeDamage(int)
- CalculateDamage(int)
- DamageShields(int)
- BypassDefence(int)
- UnitHasDied()
- MissTurn()
+ IncrementActionPoints(int)
+ DecrementActionPoints(int)
+ ToggleSelect(bool)
+ ToggleMovedThisTurn(bool)
+ ToggleAttackedThisTurn(bool)
+ ToggleMissTurn(bool)
+ ToggleWaitingToAttack(bool)
+ ToggleUnitsInRange(bool)
+ getter methods
0..*
1
AttackUnit
- unitsInRange: List<Unit>
- tilesInRange: List<Tile>
- abilityName : string
- information : string[]
+ tiles: GameObject[]
# moveSelected : bool
# attackRange: int
# actionPoints: int
- Start()
# FindUnitsInRange(Unit, int)
# AttackFlowProcess(Unit. int, int)
# AttackEnemyUnit(Unit, Unit, int, int)
# DefendAllyUnit(Unit, Unit, int, int)
# ReduceDefence(Unit, Unit, int, int)
# ReduceMultiDefence(Unit, Unit, int)
# BypassShields(Unit, Unit, int, int)
# DDoSAttack(Unit, Unit, int)
# ResetSelection()
# ResetAllTiles()
- FindUnitWithDirection(Unit, Vector3, int)
- HighlightTilesInRange(Unit, int)
- UnitHasAlreadyAttacked()
- UnitCannotMove()
- NotEnoughActionPoints()
- UnitsAreInRange()
+NoUnitsInRange()
1
Inherits
1
Associates
Tile
- matInstance : Material
+ current : bool = false
+ target : bool = false
+ selectable : bool = false
+ attack : bool = false
+ visited : bool = false
+ parent : Tile = null
+ distance : int = 0
- Start()
- Update()
- SelectTile()
- CheckTile(Vector3)
- CheckTilesInRange(Vector3, int)
+ FindNeighbours()
+ FindNeighboursInRange(int)
+ Reset();
Generic Unit Move*
- unit : Unit
- waiting : bool
- unitSelected : bool
- damage / defence* : int
+OnEnable()
+OnDisable()
+OnClickUseMove()
+DeselectMove()
+WaitToSelectUnitInRange()
*Generic Unit Move represents 1 of the 8 different child attack classes

UML Modelling – Activity Diagram

The following Diagram refers to the sequence of two players joining a game, matchmaking and then the typical sequence of events that happens until the game is finished.

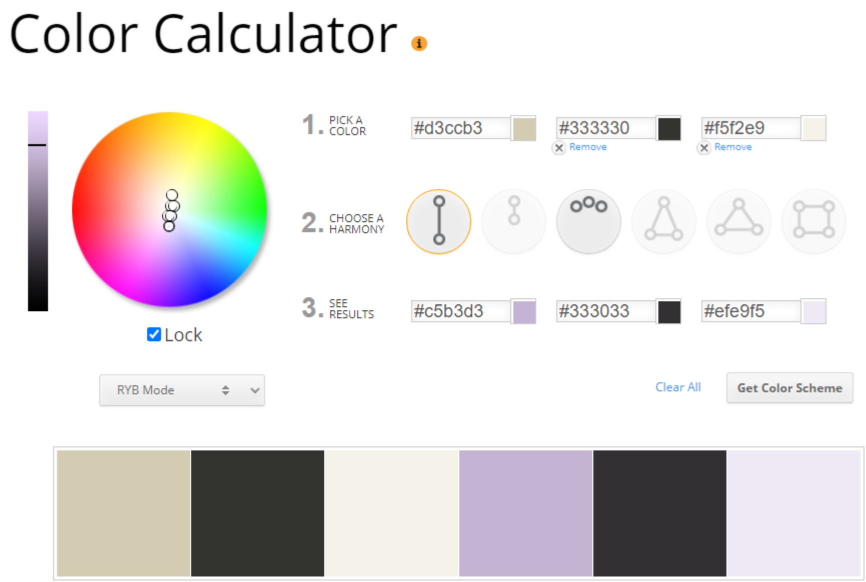
Figure X – Activity Diagram the Game



Visual Design (UIs)

To find a good colour palette for the game, I first ook inspiration from the base colour palette used in Nier Automata – a similar game with Sentient robots being a heavy theme. I then took this base colour, used a colour calculator to find harmonic colours (lighter and darker hues of the same base), and complimentary colours (opposites on the wheel) which form a good contrast.

Fig 1 – Finding an Appropriate Colour Palette [Ref]



Color Calculator

1. PICK A COLOR: #d3ccb3, #333330, #f5f2e9

2. CHOOSE A HARMONY

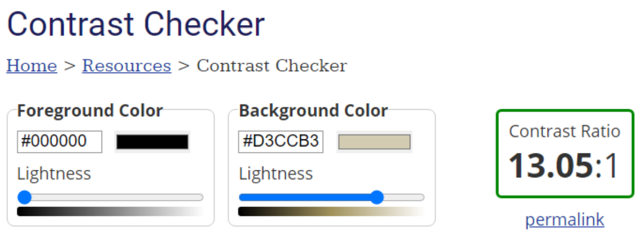
3. SEE RESULTS: #c5b3d3, #333033, #efe9f5

☑ Lock

RYB Mode

When designing the text, I made sure to make sure the text was compliment with the latest edition web content accessibly guidelines (WCAG 2.1 – published 2018 [ref] which states small text needs a minimum contrast ratio of 7:1 (level AAA). I then took the RGB hexadecimal colour codes to find that the two uses of text in my game have a contrast ratio of 18:75:1 and 13:05.

Fig – Contrast Checker for Unit Stats [ref]



Contrast Checker

Home > Resources > Contrast Checker

Foreground Color: #000000

Lightness

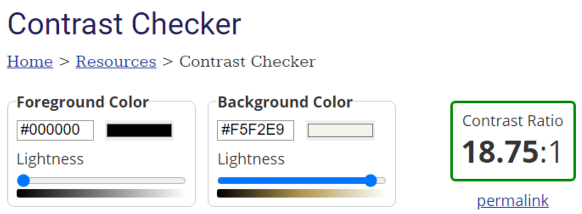
Background Color: #D3CCB3

Lightness

Contrast Ratio 13.05:1

permalink

Fig – Contrast Checker for Smaller History Log [ref]



Contrast Checker

Home > Resources > Contrast Checker

Foreground Color: #000000

Lightness

Background Color: #F5F2E9

Lightness

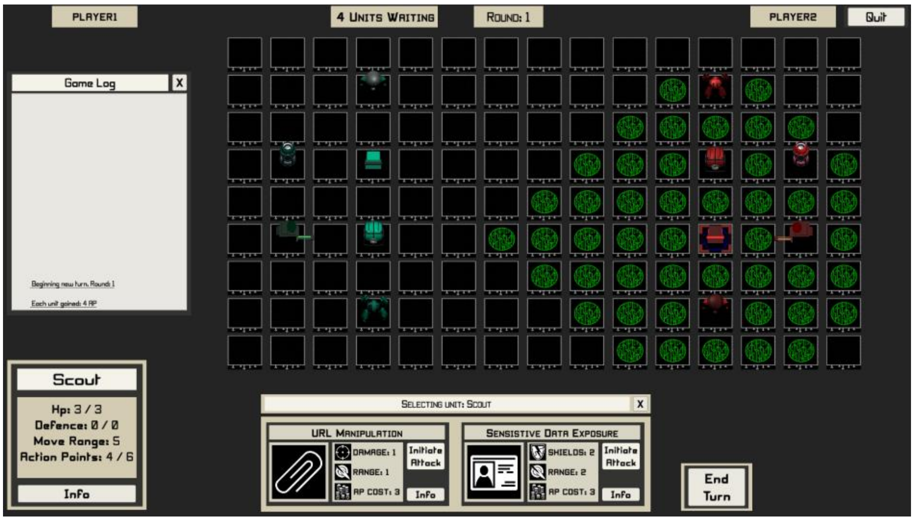
Contrast Ratio 18.75:1

permalink

Colour Blindness and Accessibility

Approximately 1 in 12 men (8%), and 1 in 200 woman (0.5%) experience some form of CVD (Colour Vision Deficiency) [ref]. Considering these numbers are so high, it is extremely important to design an easy to see game to be as inclusive and accessible as possible.

Fig 1 – Initial Colour Design



Beginning new turn. Round: 1

Each unit gained: 4 AP

Hp: 3 / 3
Defence: 0 / 0
Move Range: 5
Action Points: 4 / 6

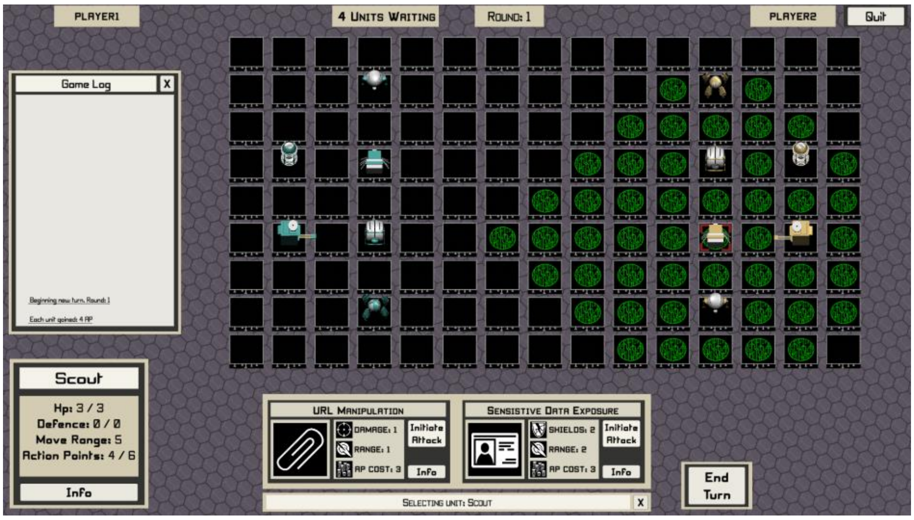
DAMAGE: 1
RANGE: 1
AP COST: 3

SHIELDS: 2
RANGE: 2
AP COST: 3

Referring the colour blindness guide [ref], I decided to use a colour palette designed for accessibility – combined with a similar colour theme to the desired UI aesthetic. This includes using higher contrasting images, avoiding red and green as a clashing complimentary colour – and also generated a patterned background as patterns can be used to clearly contrast two images/backgrounds as well. An application doodad was used to generate royalty free backgrounds with the desired palette.

This also involved changing the colour palette of the units (Red and Green) to be a lighter cyan + malt yellow (complimentary and colour blind friendly) with white overtones to contrast with the tiles better

Fig 2 – Revised Colour Design



Beginning new turn, Round: 1

Each unit gained: 4 AP

Scout

Hp: 3 / 3
Defence: 0 / 0
Move Range: 5
Action Points: 4 / 6

URL Manipulation

DAMAGE: 1
RANGE: 1
AP COST: 3

Sensistive Data Exposure

SHIELDS: 2
RANGE: 2
AP COST: 3

Selecting unit: Scout

Technical Design (Lobby Matchmaking / Hosting)

- Can talk about master /client server relationships here with a diagram
- High level design of client application / API / other servers

How Gamified Mechanics are Integrated

- As identified earlier and thus meet the requirements!
- This section will probably be quite long with multiple subsections for each mechanic and the value it adds

Cyber Security Teachings

- This section will probably be quite long with a description of types of cyber security attacks/defences identified

Mapping OWASP Attacks

The OWASP Top 10 [https://owasp.org/www-project-top-ten/]

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

Ability Name	Description	OWASP Relation
SQL Injection	Injection attacks are ranked #1 on the OWASP top 10, with SQL being the most common injection style attack. Sequel Query Language (SQL) injection exploits the interactivity between the web application and commands that query a backend database. Using SQL injection, it is possible to bypass authorisation, extract sensitive information from databases and even manipulate database entries with your own custom information.	1
XXS	Cross-Site Scripting (XSS) attacks can occur when a web application fails to validate user input by escaping special characters. As a result, an attacker can execute scripts in a victim's browser, hijack their session data and even redirect them to other phishing sites.	7
Firewall	Firewalls block and filter unwanted traffic in order to keep the web application isolated within a safe environment. Furthermore, firewalls can be used in conjunction with an Intrusion Detection/Prevention system in order to detect malicious activity and send alerts upon detection. A firewall can also be used to mitigate XML External Entities attacks (XXE)	4
DDoS	A DDoS attack involves using a network of devices in order to flood a web application, network or server with overwhelming internet traffic - and disable it from being able to handle requests.	4

	Although not directly listed in the OWASP Top Ten, Denial of Service attacks can be accomplished from exploiting XML External Entities (XXE)	
URL Manipulation	URL Manipulation changes the URL parameters in order to bypass authorisation in websites that suffer from broken access control. The goal is to access admin locations, root files and sensitive data (such as software versions) which could be used to identify vulnerabilities within the system.	5
Sensitive Data Exposure	Examples of sensitive data include personal, account and financial information. Improper handling of sensitive data in transit or storage lead to data breaches which can cripple a company's reputation. In more severe outcomes, such information could be used to commit fraud and identity theft.	3
Delete Security Logs	Insufficient data logging and monitoring allows an attacker to persist in a compromised system - undetected for a longer time without being detected. Logging pertains to recording anomalies, failed login attempts, user activity and so on, whilst effective monitoring is to know how to interpret the logged data - and how to react appropriately in the case of such detection.	10
Access Control Systems	Broken access control is present when employees/users have access to higher levels of clearance, sensitive information and possibly even "admin" operations than they require. A good access control system should be implemented once, and periodically reviewed. Control policies based on user roles should enforce the least amount of privileges each user requires to complete their interaction within the web application.	5

Unit Information

Heavy Unit

The Heavy unit specialises in disabling the operations externally (via denial of service attacks), and further disrupting internal operations (via deleting security logs).

Scout Unit

The scout specialises in exploiting broken access control systems and exposing sensitive information for very poorly designed web applications.

Analyst Unit

The analyst specialises enforcing security by patching broken access control systems, and implementing security firewalls.

Good analysts should also log and monitor internet traffic in order to detect malicious activity and react to it quickly.

Hacker Unit

The hacker specialises with direct attack vectors typical of pen-testers - that can prod a web application in search of vulnerabilities.

SQL injection and XSS scripting are two very common methods that can be brute forced in order to further exploit the web application.

Design Patterns (Unity / C#)

Singleton Design Pattern

A singleton involves creating a public static instance of a class or object such that only one version/reference of the object exists at any given time. This is very important for managers (such as the game manager, sound manager and player controller – as we do not need multiple instantiations of these objects, just one manager which can be statically called from other classes to invoke changes in the game's life-cycle)

Observer Design Pattern

The observer design pattern involves creating "listener" objects/event system which can respond to game interactions and invoke a method based on a change. For this project, the observer pattern was implemented within the UI / moves such that upon a move selected or used, the move will find the currently selected units, calculate potential targets in a nearby radius and update variables pertaining to what should be selected/highlighted – these are all temporarily cached, and then the move simply awaits a user input.

Though technically, we do not register for an event when OnEnable / OnDisable is invoked, so not a true implementation of the observer pattern.

Command Design Pattern

This pattern is extremely game-type specific and is an appropriate choice for turn – based strategies as it makes it possible to queue up actions, and reverse actions.

In essence, the player controller listens for human interactions and selects units which tells the unit controller (specific for each unit) that that unit is selected and available to move and use a move.

Furthermore, the moving itself involves a queue of tiles to movement and via the unit controller, processes each movement command until the unit has reached the target destination. More information is including about the movement later on.

The command pattern makes it incredibly easy to implement reversal of movement, however I didn't implement this feature in the game since I was designing a real time – synchronous multiplayer game (like Chess, where moves typically can't be revoked in the case of an accident). If this was a single player game, it'd be more appropriate to allow the player to reverse a move decision.

Component Design Pattern

This is the most common design pattern as Unity Development is hinged around components in the same way Java is hinged on objects. In essence, instead of having a single unit class which controls movement and control, the moves it has, the damage and stats, the buttons and UI that's linked to the unit, everything is broken down into subcomponents and stitched together.

Typically a Unit object has a Unit Script (which contains it's unit information, health, defence and movement range), and a Unit Controller Script which can access it's unit information (movement range), but sole purpose is to calculate which tiles are movable by extending of the pathfinding class (which has all the movement operations and breadth first search). Unit moves (such as SQL injection) are completely decoupled from the Unit component, and are displayed by the game UI manager which a certain unit is selected, but these moves can be switched around and reassigned to any unit

Flyweight Design Pattern

This pattern is used to save memory when many instances of the same object are created and stored. For example, each time a tile changes colour, we access the material component and change the colour of the material. If we assigned this a new material colour, we could accidentally create a copy of the material where all the old colour materials will still be stored in memory (despite not even ever being reused or shown), which for a 100 tiles with 5 different colours would quickly lead to loads of excess tile-material objects existing. By using the profiler, we can access the material count and material memory used (**testing**). A better solution is we assign a single instance of a material for each tile (upon awake /object creation), and we **update** this instance (and not create a new material every time we change colour). Although this sounds like common sense, it is easy to miss!

A Justification of this Approach

- Include some alternative approaches here for both applications (tablet/mobile/desktop/web) and server hosting

- Also SQL/Database/account registration if we get to that stage

Implementation of Project

Development Environment (Unity3D)

Unity3D was chosen as an appropriate development environment because ultimately I wanted to design a game that could be exported to the web which Unity3D has seamless integration for (WebGL). I did consider Phaser3 / HTML as an alternative platform which could progress my web development skills, and integrate this with NodeJS. However, Phaser3 is still a relatively small open source project, whereas Unity3D is a much more widely used industry trade skill – with ample resources and supporting frameworks online.

Furthermore, working with Unity allowed me to hone my understanding of C# which is very syntactically similar to Java, and share a lot of similarities such as being statically typed object-oriented languages; a programming style I am very familiar with now.

Client – Server Multiplayer

Photon Unity Networking 2 (PUN2) is a free to use API which allows developers to host a remote server in which clients can connect to. PUN2 was chosen because it has an ample tutorial base, as well as very strong integration with unity and even support for cross platform (which would make exporting to the web later very easy). PUN2 is typically a paid service, but offers developers free server hosting up to 20 concurrent players, which for the purposes of this research perfect, is more than adequate!

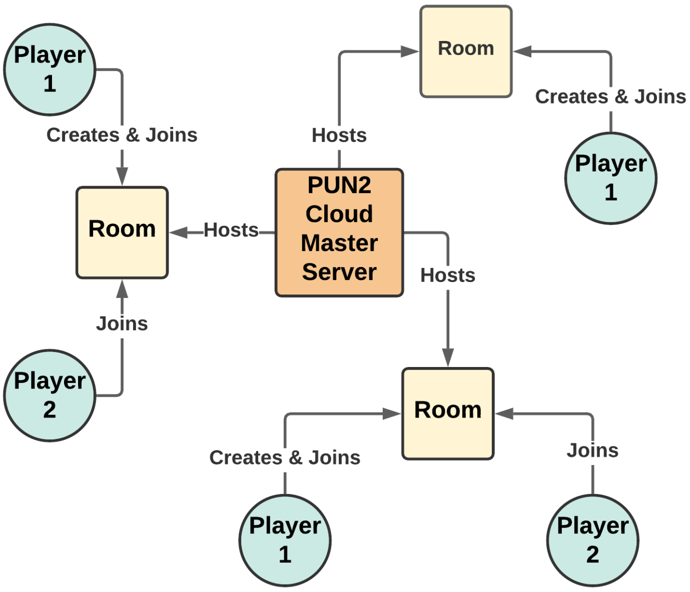
Mirror is an alternative free and open source library which was built to replace Unity's previous deprecated multiplayer libraries (UNET). Although Mirror is completely free to use, relatively simple and highly covered, there were limitations the server and client are **one** project, which would mean that I would either have to run my version of the game constantly to host multiplayer sessions – or develop a server to host game sessions. Thankfully PUN2 makes this process seamless and easily such that other people can participate in my game without my involvement at all.

Photon Unity Networking 2 Architecture

Commented [RB5]: In this section, we could reference the master-server architecture and how it fulfils an initial requirement of being a readily available web-based game

PUN2 follows a classic Server – Client architecture; PUN itself hosts cloud servers around the globe in which local clients can connect to remotely. It is free to host 20 concurrent users on any master server, however you'll need to pay to increase that capacity.

When a client connects to the PUN Network initially, it invokes the 'JoinOrCreateRoom()' function which essentially tries to join a non-full room (if it exists), or create one if there are no other rooms (with space) available. The first player (client) to do this becomes the **master client** which is responsible for hosting the game, and for my game, is player 1. If the master client disconnects for any reason, there are options for the next available client (if present) to be promoted to a new master client, however since my developed project is only 2 players, it makes sense just to return the other player to the main menu and end that room session. The biggest number of rooms I can host at one time is therefore 10 (supporting 20 connected clients concurrently).



[Diagram reference: ref – TODO: sketch this myself but for 2 people, but still reference!]

Remote Procedure Calls RPCs

PUN RPCs are a core part of the multiplayer aspect in which a client can communicate the position or state of his game directly to all other clients.

Fig X – Illustrating Sending a Method Update to a selected user

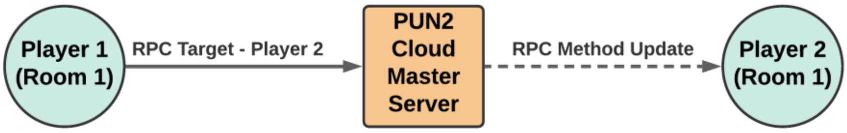
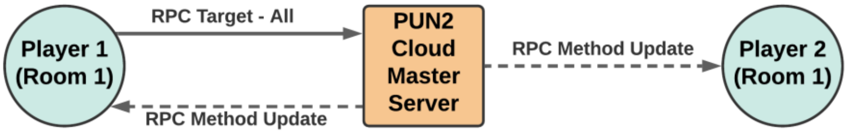


Fig X – Illustrating Sending a Method Update to all users within the room



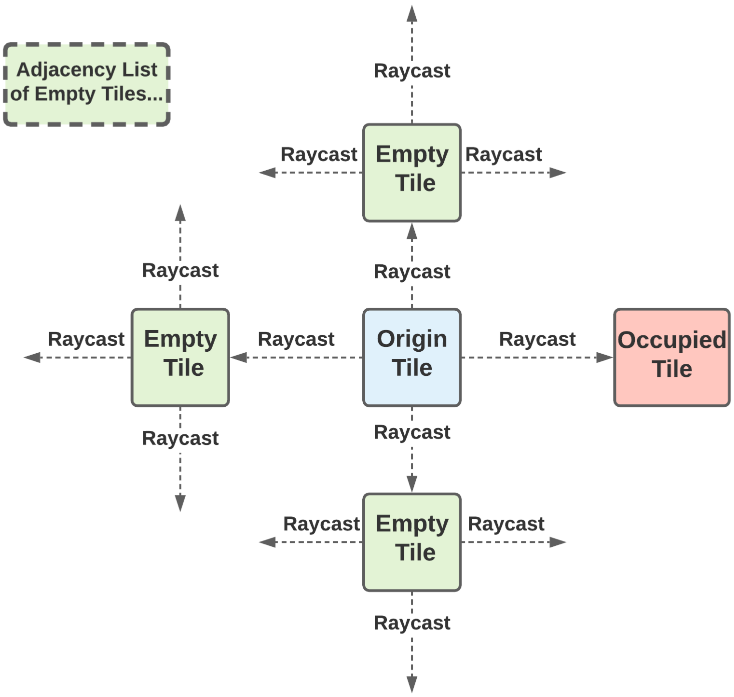
Commented [RB6]: Can include take damage RPC as a direct example of its use

[Insert diagram similar to this, but to illustrate how a client creates and sends an RPC call] For example, when a new turn is loaded, the client of the previous turn will call photonView.RPC("InitiateNextTurn", RpcTarget.ALL); where the photon view is the ID of the player/client invoking the message, InititateNextTurn is the function the client is telling all other clients to run, and RpcTarget.ALL represents all other connected clients. For larger games, it's possible to pass in the client's particular photon ID here such that you only communicate with one client, and not all.

Furthermore, if the master client who was to begin the game was to invoke photonView.RPC("BeginGame", RpcTarget.Allbuffered); can apply, where buffered instructs the call to wait and ensure **all** clients will receive the message even if they haven't fully loaded the game scene (in the case of delayed connections).

Movement Mechanics – Breadth First Search with Shorted Path

Fig X – Scanning adjacent tiles with a ray cast, and adding to the adjacency list



In order to calculate the pathfinding movement possibilities of each unit on the tile map, the player controller inherits a pathfinding class which utilises an implementation of breadth first search. BFS with shorted path is used to calculate all the possible tiles a unit can move to within the maximum movement distance range specified by the Units properties.

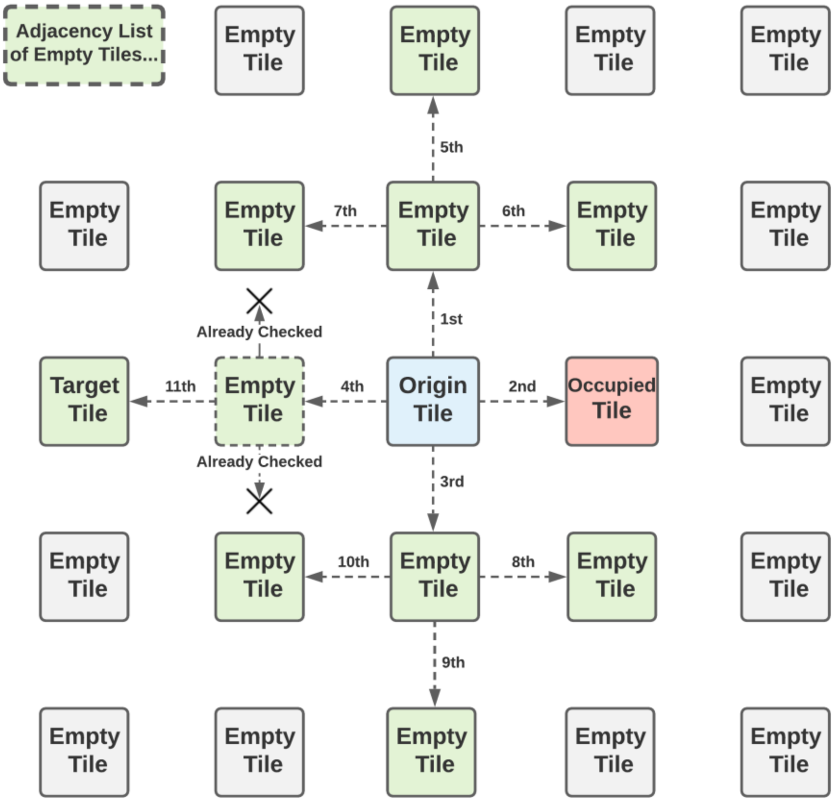
At the end of the selected tile, the algorithm returns the shorted path to move to that location tile within range. Essentially, the current tile the unit on is processed first, then each neighbouring node outwards (in all directions) is processed (if applicable) and only processed once – in a breadth first search like manor. [Tutorial here]

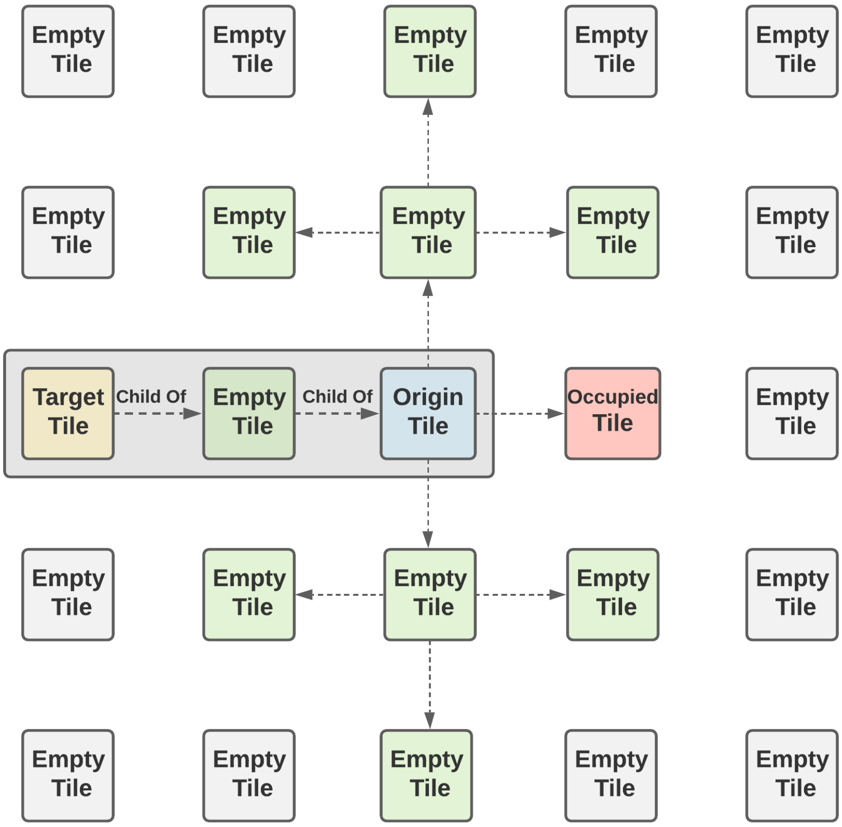
1. At the start of runtime, all tiles (nodes) on the board are located and cached to save performance during calculations later on
2. When a unit is selected, a ray is cast from the unit's location to determine the tile/node below it
    - From this tile / node, all neighbours (left, right, up, down) are calculated and added to adjacency list

3. We then initiate breadth first search to find all selectable tiles:
    - We first enqueue the current tile to a Queue of tiles that need to be explored
        - We set this tile visited to true so that we never recheck/come back to this tile
    - We then iterate through the Queue all while it is not empty
        - If this ever reached 0, then we would be in a position where we had no legal moves (surrounding by a block in all 4 movement directions)
        - We add that tile to a list of selectable tiles, and change that colour
        - After processing that tile, we add all neighbours in the adjacency list to the queue for further processing
            - If a tile in that list has already been marked as visited, we do not this to the queue for processing
            - For each tile in this adjacency list, we set the parent node of this tile as the current tile before it, that way we can backtrack a path to the destination later on if needed
            - We then mark this child node as visited, as set its distance as 1 + the distance of the previous tile

Commented [RB7]: ▪Note BFS here is used just to find selectable tiles, not a path to that tile

Commented [RB8]: This distance is important so that we can stop running BFS as soon as this distance is reached as specified by the maximum movement distance of the unit


Adjacency List of Empty Tiles...
Empty Tile
Empty Tile
Empty Tile
Empty Tile
5th
Empty Tile
Empty Tile
7th
Empty Tile
6th
Empty Tile
Empty Tile
Already Checked
1st
Target Tile
11th
Empty Tile
4th
Origin Tile
2nd
Occupied Tile
Empty Tile
Already Checked
3rd
Empty Tile
Empty Tile
10th
Empty Tile
8th
Empty Tile
Empty Tile
9th
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile


Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Target Tile
Child Of
Empty Tile
Child Of
Origin Tile
Occupied Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile
Empty Tile

Testing Strategy and Results

ParallelSync for Quick Builds / Debugging

ParallelSync is a small open source extension for Unity3D which allows you to clone the Unity development environment and run multiple editors of the same project (which means you do not have to build and run the game every time to test 2 or more players!). This extension has proven to be an invaluable timesaver for the development and quick testing of multiplayer development. ParallelSync works by cloning the structure of the project without needing to duplicate and load all files/assets/game states, instead it establishes pointers to the original files/assets and game scenes which have read access only (and do not affect the state of the master development environment). Because of this, it is easy to create, load and destroy as many clone environments as required.

- Need to research multiple methods on how to test an application
- https://www.softwaretestinghelp.com/types-of-software-testing/

JUnit Test Cases

- Write a test case that simulates the walkthrough of client connecting (loading game, taking turn, winning, registering, score updating, achievements being earned etc)
- Other general backend cases, version control is another one! (compatibility)

Integration Testing

- Especially for Networking feature
    - Identify all edge cases and ensure checks are valid for each one (include a state machine with transition!)

UI / Interface Testing Testing

- Testing availability functionality / buttons etc with a table of all edge CASES

Performance Testing

- Since Unity is still a relatively new tool to me, it was important to make a silly mistake with over reliance on any of the networking / continual update hugs
- Profile Analyser
- To review performance analysis and identify performance hogs / what needs to be optimised
- **Loading testing, tho we only have 20 CCU limit anyway**

One of the bigger issues was how to go about discovering tiles and stuff, and what I had noticed is my update loop to check tiles was called each instance, and as such was sending out many raycasts per frame (unnecessarily)

A more efficient method would have been to assign all tiles on the map in a 2D array, with the coordinates of each unit at each tile, and probably make unit a child of that tile such that when iterating through the tiles in a map, if the tile has a child, then it's not movable, or the unit is attacking. However, this maths greatly complicates 3D planes (e.g. climbing up/down a block, not just a 2D array), as well as combines the components together in a less decoupled way. As a result I still decided on using raycasts as it's performance impact wasn't that great, and it allows greater flexibility as a simple raycast can be used for move detection/blocking, any new tiles being dynamically added (not having to manually assign a 2D array coordinates), map changes, and even map elevation changes – which is present in Tactics stile movement games. Though in this case, I eventually decided to omit having map changes/elevation in favour of a simpler playing field. Again one of the requirements of this project was scalability and flexibility for expansion which my solution does offer

Profiler adds a little overhead to the game but can find computationally/memory heavy processes by adding a marker before the beginning of every function call (method update etc)

Initially I was getting the component reference each time I was updating the tile – which would lead to a massive garbage disposal (in the GC) – which would cause occasional lag spikes as it became too full and needed to be cleared, but I have since fixed that by getting the reference once, and caching it so it can used over multiple frames. Unfortunately there is still data lost whenclicking a tile to find moves due to the path finding algorithm which will create a lot of lists/queses to iterate through adjaceny lists and find all valid paths within a distance. However, this isn't too bad of a tradeoff since the MS time requirement is really minimal (se value!)

An alternative for tiles checking in update is to have event listeners, however this isn't that costly at all,

Unit Testing

Benefits of unit testing in unity

Typically testing is very easy vbia playmode, however it can take a long time to write code, switch to unity, launch the game, connect to multiplayer, get to a certain state, see the stare of one change, then repeat if it's not fixed. A much quicker test is automated tests on functions which should always output the same result on a given input. The following are some areas of testing

User Interaction / Feedback Testing / Acceptance Testing

- Double check that this doesn't require ethics approval and is a valid form
- **Usability testing**

Exploratory Testing

- Explore through certain aspects of the application, cleaning up redundant processes, modularising the project as much as possible and keep a track of these

A Critical Evaluation

Evaluation of Approach

- What other alternatives could've been done
- What other APIs / thingymabobs could've been used
- Have all the MoSCoW tasks been achieved?

Evaluation of Final Implementation

- What gamified mechanics have been implemented
  - What mechanics are missing that ideally would've been included given more time
- Does the project meet all the functional and nonfunctional requirements?
- What needs more time or development?
- Is it good on multiple devices as a web application, i.e tablet with touch based response, or just good on a desktop scenario?
- Are there any other features that could have been added to improve the overall complexity of the project
- Comparison to **other online similar projects / prototypes / research**

User Feedback Evaluation

- If time permits, write a pre-analysis questionnaire then get the user to experience the game, then write a post-analysis questionnaire
- Collect some qualitative feedback about overall impressions
- Is the usability aspect good?

Evaluation of Security

- Using PUN vs Mirror has it's security drawbacks, is the application easy to break and cheat? Though this shouldn't be an issue given the light hearted design, like no one would bend heaven and earth to cheat in monopoly?

Evaluation of Testing Strategies

- Are they enough / sufficient?
- Are there any other testing strategies that could've been done given more time?

EXTRA NEW POINTS

- Other frameworks, extensions especially the framework that has iterations to complete later on – and builds upon the MDA because gamifying games is hard

Conclusions and Future Work

My Contribution

My contribution is adding a new turn based strategy game that can be played by 2 people – that isn't necessarily just for training a business. A lot of cyber security games were multiplayer for a business (or singleplayer simulations), or singleplayer for fun (e.g. young kids/students) **but** not multiplayer and relevant for both, whereas this game can potentially fulfil both of this criteria.

Furthermore, this game serves as a solid framework in which more turn based games, scenarios, units and moves can flexibility added whereby only the aesthetics / animations and physical designs need to be altered.

What else??

Project Management

- Which tools were the most important
    - What could've been used instead / done differently
- Which techniques were the most important
    - What could have been changed

Reflection

- Risk Assessment
    - Any extra risks / unforeseen stuff / changes?
- Gantt Charts
    - Overall structure / progress? Start anything earlier? More time for exams!!!

Future Additions

Conclusion

-

Bibliography

Appendices